

REMARKS

Claims 1 – 25 are pending in the application. Claims 1 – 25 have been rejected. Claims 1 – 14, 16 – 19 and 21 – 24 have been amended.

The title has been amended to address the examiner's objection.

Claims 1, 14 and 15 stand rejected under 35 U.S.C. 101. More specifically, the examiner has set forth that claims 1, 14 and 15 are not limited to “a practical application of an abstract idea which produced a useful, concrete, and tangible result”. (Office Action, page 2). This rejection is respectfully traversed. When discussing patentable subject matter for computer related inventions, the MPEP sets forth

The claimed invention as a whole must accomplish a practical result. . . . The purpose of this requirement is to limit patent protection to inventions that possess a certain level of “real world” value, as opposed to subject matter that represents nothing more than an idea or concepts, or is simply a starting point for future investigation or research. . . Accordingly, a complete disclosure should contain some indication of the practical application for the claimed invention, i.e., why the applicant believes the claimed invention is useful. (M.P.E.P. 2106A, *citations omitted*.)

Applicants respectfully submit that claims 1, 14 and 15 are directed to the useful result of “integrating data between a plurality of software applications”.

Claims 1, 14 and 15 stand rejected under 35 U.S.C. 101 based on the theory that these claims are not directed to a statutory class. This rejection is respectfully traversed. Claim 1 has been amended to specifically claim a “system”. Additionally, claim 14 is directed to a “factory system” and claim 15 is directed to a “domain application”. Each of these claims includes elements that set forth the structure of the system, factory system or application as the case may be. Accordingly, these claims define a useful machine as prescribed by identifying the physical structure of the machine.

Claims 16 stands rejected under 35 U.S.C. 101. More specifically, the examiner has set forth that claim 16

is directed to a method for integrating data between a plurality of software applications comprising providing a domain object superclass, providing first-level sub-classes of the domain object superclass, providing a service and providing and performing an operation related to the domain object. These method steps do not appear to have practical applications which produce useful, concrete, and tangible results on the State Street Formulation. (Office action, Pages 3 and 4.)

This rejection is respectfully traversed. Applicants respectfully submit that claim 16 is directed to the useful result of “integrating data between a plurality of software applications”.

Claims 1 – 25 stand rejected under 35 U.S.C. 112, second paragraph.

More specifically, the examiner questions the meanings of the terms “factory environment,” “factory system” and “factory.” A “factory system” is defined a single system that control product flow and equipment uses within a “factory”. (See e.g., application page 2, lines 17 – 13.) A factory environment is the environment in which the factory system is instantiated. The factory is a physical manufacturing factory which is controlled via the factory system.

Additionally, the examiner questions the meaning of the term “architecture.” This term has been removed based upon the claim amendments.

Additionally, the examiner questions whether the terms “domain object superclass” and “domain object” are the same. The specification sets forth that an object is an instance of a class, which provides a template for the object. (See e.g., application page 7, lines 22 – 25.) The specification further sets forth that a class has a position within a hierarch and that methods or code in one class can be inherited from a superclass. (See e.g., application page 8, lines 11 – 16.) The specification further sets forth that a domain object super class is an abstract class, domain objects are not an instantiation of the domain object superclass. Rather, sub classes of the domain object superclass are provided in the domain object framework and domain objects are instantiation s of these subclasses. (See e.g., application page, 10, lines 15 – 19.) Accordingly, a domain object and a domain object superclass are not the same term.

Additionally, the examiner sets forth that it is unclear with what entity the domain object communicates and how significant the limitation “as if the operation were performed on the domain object” is. The specification sets forth that each integrated application to which a

component provides an interface is also capable of implementing the subclasses of domain object superclass to implement the functionality of the domain object that the integrated application understands. Code from the integrated application is used to perform the operation without the need to revise the calling domain application to use the integrated application. (See e.g., applicaitn page 12, lines 7 – 18.) This functionality is claimed via the “as if the operation were performed on the domain object” language of the claim.

Claims 16, 17, 18, 19, 21, 22, 23 and 24 stand rejected under 35 USC 112, second paragraph based upon various antecedent basis issues. Claims 16 – 19 and 21 – 24 have been amended to address these rejections. However, it is respectfully submitted that claims 22 provides antecedent basis support for the term “application-specific domain object” at claim 22, line 5.

Claims 1, 14, 15, 16 and 21 stand rejected over Baxter, U.S. Patent No. 6, 289,500 (Baxter).

The present invention, as set forth by independent claim 1, relates to a system for integrating data between a plurality of software applications in a factory environment. The system and the plurality of software applications are stored on computer readable media. The system includes a factory system and a domain application. The factory system includes a domain object superclass, a plurality of first-level subclasses of the domain object superclass, an instantiation of one of the plurality of first-level subclasses corresponds to a domain object, the domain object represents an item in a factory, and a service. The service provides an operation related to the domain object. The service comprises at least one component. Each of the at least one component is operable to perform the operation related to the domain object. The domain application includes an implementation of one component of the at least one component of the service of the factory system to perform the operation related to the domain object.

The present invention, as set forth by independent claim 14, relates to a factory system for integrating data between a plurality of software applications in a factory environment. One of the plurality of software applications corresponds to a domain application. The factory system includes a domain object superclass, a plurality of first-level subclasses of the domain object superclass, an instantiation of one of the first-level subclasses of the plurality of first-level

subclasses corresponds to a domain object, the domain object representing an item in a factory and a service, the service providing an operation related to the domain object, the service comprising at least one component, each of the at least one component corresponding to operable to perform the operation related to the domain object wherein the domain application includes an implementation of one component of the at least one component of the service of the factory system to perform the operation related to the domain object.

The present invention, as set forth by independent claim 15, relates to a domain application for integrating data between a plurality of software applications in a factory environment. One of the plurality of software applications corresponds to a factory system. The factory system includes a domain object superclass, a plurality of first-level subclasses of the domain object superclass, an instantiation of one of the first-level subclasses corresponding to a domain object; the domain object represents an item in a factory and a service. The service provides an operation related to the domain object using a component. The service comprises at least one component. Each of the at least one component is operable to perform the operation related to the domain object. The domain application includes an implementation of one component of the at least one component of the service of the factory system, wherein the component is operable to perform the operation related to the domain object.

The present invention, as set forth by independent claim 16, relates to a method for integrating data between a plurality of software applications in a factory environment which includes providing a domain object superclass in a first software application. The first software application corresponds to a factory system, providing a plurality of first-level subclasses of the domain object superclass, instantiating one subclass of the plurality of first-level subclasses to create a domain object, the domain object representing an item in a factory, and providing a service that provides an operation related to the domain object, the service comprising at least one component, each of the at least one component being operable to perform the operation related to the domain object, performing the operation related to the domain object using an implementation of one component of the at least one component of the service, the implementation being provided by a second software application, the second software application corresponding to a domain application.

The present invention, as set forth by independent claim 21, relates to a computer program product for integrating data between a plurality of software applications in a factory environment which includes instructions for providing a domain object superclass in a first software application, the first software application corresponds to a factory system, instructions for providing a plurality of first-level subclasses of the domain object superclass, instructions for instantiating one subclass of the plurality of first-level subclasses to create a domain object, the domain object representing an item in a factory, instructions for providing a service that provides an operation related to the domain object, the service comprising at least one component, each of the at least one component being operable to perform the operation related to the domain object, and instructions for performing the operation related to the domain object using an implementation of one component of the at least one component of the service, the implementation being provided by a second software application, the second software application corresponding to a domain application, and a computer-readable medium for storing the instructions for providing the domain object superclass, the instructions for providing the plurality of first-level subclasses, the instructions for instantiating, the instructions for providing the service, and the instructions for performing the operation.

Baxter discloses a mechanism for instantiating domain neutral objects with suitable domain specific run time extensions in an appropriate collection. Baxter further discloses as an example of an extension a domain specific extensible item factory. Baxter sets forth that the domain specific extensible item factory is a standard factory object that creates all extensible items and any owned extensions in the default collection. Before the object creates the extensible item, the object looks for a special factory to which the object can delegate the extensible item. Baxter sets forth that there is only need for one extensible item factory. (Baxter Col. 11, lines 1 – 5).

Baxter does not teach or suggest a system for integrating data between a plurality of software applications in a factory environment much less such a system which includes a factory system and a domain application where the factory system includes a domain object that represents an item in a factory and a service that provides an operation related to the domain object, all as required by claim 1. Accordingly, claim 1 is allowable over Baxter. Claims 2 – 13 depend from claim 1 and are allowable for at least this reason.

Baxter does not teach or suggest a factory system for integrating data between a plurality of software applications in a factory environment, much less such a factory system which includes a domain object superclass, a plurality of first-level subclasses of the domain object superclass, an instantiation of one of the first-level subclasses of the plurality of first-level subclasses corresponds to a domain object, the domain object representing an item in a factory and a service, the service providing an operation related to the domain object, the service comprising at least one component, each of the at least one component corresponding to operable to perform the operation related to the domain object wherein the domain application includes an implementation of one component of the at least one component of the service of the factory system to perform the operation related to the domain object, all as required by claim 14. Accordingly, claim 14 is allowable over Baxter.

Baxter does not teach or suggest a domain application for integrating data between a plurality of software applications in a factory environment, one of the plurality of software applications corresponds to a factory system, the factory system includes a domain object superclass, a plurality of first-level subclasses of the domain object superclass, an instantiation of one of the first-level subclasses corresponding to a domain object; the domain object represents an item in a factory and a service, the service provides an operation related to the domain object using a component, the service comprises at least one component and each of the at least one component is operable to perform the operation related to the domain object, much less such a domain application which includes an implementation of one component of the at least one component of the service of the factory system, wherein the component is operable to perform the operation related to the domain object, all as required by claim 15. Accordingly, claim 15 is allowable over Baxter.

Baxter does not teach or suggest a method for integrating data between a plurality of software applications in a factory environment which includes providing a domain object superclass in a first software application, the first software application corresponds to a factory system; providing a plurality of first-level subclasses of the domain object superclass; instantiating one subclass of the plurality of first-level subclasses to create a domain object, the domain object representing an item in a factory; providing a service that provides an operation related to the domain object, the service comprising at least one component, each of the at least

one component being operable to perform the operation related to the domain object; and, performing the operation related to the domain object using an implementation of one component of the at least one component of the service, the implementation being provided by a second software application, the second software application corresponding to a domain application, all as required by claim 16. Accordingly, claim 16 is allowable over Baxter. Claims 17 – 20 depend from claim 16 and are allowable for at least this reason.

Baxter does not teach or suggest a computer program product for integrating data between a plurality of software applications in a factory environment which includes instructions for providing a domain object superclass in a first software application, the first software application corresponds to a factory system; instructions for providing a plurality of first-level subclasses of the domain object superclass; instructions for instantiating one subclass of the plurality of first-level subclasses to create a domain object, the domain object representing an item in a factory; instructions for providing a service that provides an operation related to the domain object, the service comprising at least one component, each of the at least one component being operable to perform the operation related to the domain object; and instructions for performing the operation related to the domain object using an implementation of one component of the at least one component of the service, the implementation being provided by a second software application, the second software application corresponding to a domain application; and, a computer-readable medium for storing the instructions for providing the domain object superclass, the instructions for providing the plurality of first-level subclasses, the instructions for instantiating, the instructions for providing the service, and the instructions for performing the operation, all as required by claim 21. Accordingly, claim 21 is allowable over Baxter. Claims 22 – 25 depend from claim 21 and are allowable for at least this reason.

CONCLUSION

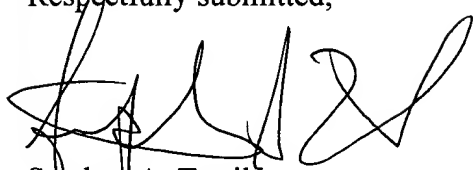
In view of the amendments and remarks set forth herein, the application is believed to be in condition for allowance and a notice to that effect is solicited. Nonetheless, should any issues remain that might be subject to resolution through a telephonic interview, the examiner is requested to telephone the undersigned.

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on October 18, 2004.


Attorney for Applicant(s)

10/18/04
Date of Signature

Respectfully submitted,


Stephen A. Terrile
Attorney for Applicant(s)
Reg. No. 32,946